Ruby - Bug #18169

Local copies of gemified libraries are being released out of sync with their gems

09/15/2021 07:41 PM - headius (Charles Nutter)

Status:	Closed		
Priority:	Normal		
Assignee:	hsbt (Hiroshi SHIBATA)		
Target version:			
ruby -v:		**Backport:**	2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN

Description

The CRuby codebase includes a number of libraries that have been gemified, more and more with each release. Unfortunately, these libraries are continually out of sync with both their home repositories and their released gems.

The problem lies in the fact that CRuby keeps a local copy of these libraries within the CRuby git repository, and allows committers to make changes either in the CRuby repository or in the gem's home repository. This has led to many releases of Ruby shipping code that **does not correspond to any released version of the related gem**.

I have filed several issues about this but the root cause has not been addressed:

- https://github.com/ruby/ostruct/issues/11
- https://github.com/ruby/matrix/issues/12
- https://github.com/ruby/prime/issues/11
- https://github.com/ruby/webrick/issues/48
- https://github.com/ruby/rdoc/issues/835
- https://github.com/ruby/rexml/issues/79
- https://github.com/ruby/fileutils/issues/59
- https://github.com/ruby/ostruct/issues/31

If these gems are to live on their own as standalone libraries/gems then one of two things must happen:

- All changes to them must go into their repositories. This would be the cleanest option. CRuby would, like JRuby, source these libraries directly from released gems, and no copies of their sources would be versioned in the CRuby git repository.

OR

- CRuby-local changes to these libraries must be prohibited from being released unless there is a corresponding gem release. This would require automated or manual auditing at release time, to ensure that the claimed gem version actually corresponds to the sources being shipped.

In addition to making it impossible for external users of these gems to match CRuby releases, there are more serious implications:

- These hybrid intra-version changes to these libraries cannot be audited to a specific gem release. This could affect stability and security when users attempt to sync their local gem sets to the ones that shipped in a given version of Ruby.
- Security fixes have gone out in CRuby releases but no corresponding x.x.y or x.x.x.y security release of the gem was released. This prevents users from fixing the security issue locally without either upgrading CRuby or also including new functionality changes (which may or may not work on the current version of Ruby).

See the rexml issue above for one example of the security problem. Until the gem was released, it was not possible to install any gem version with the security fix without upgrading functionality elsewhere in rexml.

I believe it is time for CRuby to stop making changes to gemified libraries directly in the CRuby repository. These libraries have their own gems, repositories, and issue trackers, and that is where they should be maintained.

Related issues:

Related to Ruby - Misc #16778: Should we stop vendoring default gems code?	**Rejected**
Related to Ruby - Feature #19351: Promote bundled gems at Ruby 3.3	**Closed**

History

#1 - 09/15/2021 07:48 PM - headius (Charles Nutter)

I am unable to edit the description, but I wanted to make clear that the two proposed solutions are mutually exclusive, and I **strongly** recommend that the gem sources be removed from the CRuby repository rather than auditing them at release time.

#2 - 09/15/2021 07:55 PM - headius (Charles Nutter)

- Description updated

#3 - 09/15/2021 11:01 PM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned

- Assignee set to hsbt (Hiroshi SHIBATA)

#4 - 09/16/2021 03:36 AM - headius (Charles Nutter)

I am available to help make this happen or describe how we are doing it in JRuby. The short description is just that we install to a temporary location and copy everything in place as part of the build. However, we do that by bootstrapping with an existing version of JRuby (self-hosted style). For CRuby, if we do not want to depend on an existing install of Ruby, perhaps it would be better to manually fetch and unpack the specific gem versions into stdlib during an early phase of the build?

I can be pinged on Slack or on our Matrix #jruby channel.

#5 - 09/16/2021 03:37 AM - headius (Charles Nutter)

An alternative might be to check out the repositories at a specific tag and copy them in place, rather than going to the gems.

#6 - 09/16/2021 12:09 PM - hsbt (Hiroshi SHIBATA)

We are sorry to confuse about the versioning policy for the default gems in our transition phase.

I only maintain the version of the default gems with the new release version of Ruby like 2.6.0.
I forgot them with stable version release like 2.6.1, 2.7.1 and more.

I and usa discussed them.

- We should bump version like fileutils-1.4.1.1 or 1.4.2 when releasing the new stable version in Ruby repo.
- We should backport this changes to the upstream repo in github.
  - It's good to release the same version of default gems. But It's maintainer's convenience.

I think JRuby have the several options without the release version of default gems.

1. Copy the default gems from ruby_2_6 branch, don't care of versioning.
2. Merge from the latest version of the default gems like fileutils-1.5.0 skipped 1.4.1+2.6 changes.
3. etc.

#7 - 09/16/2021 02:50 PM - headius (Charles Nutter)

> We should backport this changes to the upstream repo in github.

Why not make the changes to the gem itself and be certain that all changes are already in the canonical repository?

If it is a problem of testing in CRuby, you can release prerelease gems until the changes have stabilized, and have CRuby install those during the build.

I do not see the benefit of having two repositories for every default gem and being forced to track both and sync both. It only seems to cause problems.

Why is a copy of the default gems maintained in the CRuby repository? If you can explain that to me, perhaps we can come up with a better solution.

> Copy the default gems from ruby_2_6 branch, don't care of versioning.

We will only install the gems from released versions for the reasons I mention in the description. When we have local copies of the files, we get bug reports and pull requests against them, and they clutter up the repository with files we do not control and do not maintain.

> Merge from the latest version of the default gems like fileutils-1.5.0 skipped 1.4.1+2.6 changes.

Is there a listing of the supported Ruby versions for these newer gem releases? We are reluctant to use newer versions of the gems because there's no information about compatiblity. When we ship a JRuby that is compatible with 2.6.8 or 2.7.x we would prefer to ship the exact same version of the default gems.

#8 - 09/16/2021 11:44 PM - hsbt (Hiroshi SHIBATA)

- Related to Misc #16778: Should we stop vendoring default gems code? added

#9 - 09/17/2021 12:06 AM - hsbt (Hiroshi SHIBATA)

> Why not make the changes to the gem itself and be certain that all changes are already in the canonical repository?
> (snip)
> I do not see the benefit of having two repositories for every default gem
> Why is a copy of the default gems maintained in the CRuby repository?

Because you are not develop CRuby. We need to test EVERY commits of CRuby with default gems. If CRuby or stdlib broke the test/CI, We should investigate and fix them.

I think its differences are ownership of stdlib. We are developing stdlib and CRuby. I and mame investigate git worktree, submodule and the current workflow of CRuby. The current git cli can't handle 2+ of git repos easily.

I'm really tired to discuss about this. We already have a long discussion with https://bugs.ruby-lang.org/issues/16778.

I only coordinate the versioning issues, not our workflows.

#10 - 09/17/2021 02:44 PM - Dan0042 (Daniel DeLorme)

hsbt (Hiroshi SHIBATA) wrote in #note-9:

> Because you are not develop CRuby. We need to test EVERY commits of CRuby with default gems. If CRuby or stdlib broke the test/CI, We should investigate and fix them.

If I understand correctly, you mean that it's hard/impossible to test the gemified libs via CI unless they are part of the ruby monorepo? Is that the main difficulty?

#11 - 09/18/2021 02:48 AM - hsbt (Hiroshi SHIBATA)

Dan0042 (Daniel DeLorme) wrote in #note-10:

> If I understand correctly, you mean that it's hard/impossible to test the gemified libs via CI unless they are part of the ruby monorepo? Is that the main difficulty?

Yes. Ex. "ruby/setup-ruby" only provide the binary of ruby-head with every day. We can find the CI failure after 1+ days.

And We have over the 70+ repositories as default gems. It's difficult to monitor the all of CI status for me (and core developers.) now.

We try to migrate stdlibs to shipped gems step by step includes CI environment, toolchains and release workflow.

#12 - 09/23/2021 11:55 PM - headius (Charles Nutter)

> I only coordinate the versioning issues, not our workflows.

Ok, who should I talk to about fixing the workflow?

I proposed two options for you:

- Remove the sources from the CRuby repository

OR

- Add some process to guarantee a Ruby release never goes out with unreleased gem sources

You have made it clear that Ruby is either unable or unwilling to remove the sources from the repository, which then leaves only the second option.

I sympathize with your situation. You currently have a workflow that requires the gem sources to be present immediately, and do not wish to add the overhead of cloning those repositories before running builds or testing in CI. Unfortunately this workflow also has the side effect of causing unreleased gem sources to make it into CRuby releases.

This is still a major problem that needs to be fixed.

Today I started updating JRuby's stdlib to 3.0 and found more gems that are out of sync. Is there any workflow in place to prevent this from happening? It seems to happen every time CRuby does a release.

- debug: https://github.com/ruby/debug/issues/298
- cgi: https://github.com/ruby/cgi/issues/10

These are just two that I happened to look into because we have patched them for JRuby in the past. There are likely others that I did not inspect.

#13 - 09/23/2021 11:56 PM - headius (Charles Nutter)

> Today I started updating JRuby's stdlib to 3.0

That's 3.0.2 specifically.

#14 - 09/24/2021 05:14 PM - enebo (Thomas Enebo)

Perhaps a possible solution is when anyone edits the version of a default gem within CRuby they change the version of that gem to reflect it has not been released. For example, if cgi-3.0.2 needs changes on first change make it 3.0.3-dev (in other words no gem authors should be putting a final release for anything other than synchronization of gems at the point of a CRuby release). Release manager then will have an obvious marker that those gems have not been updated with the external gem source. Once synced they remove the -dev.

I can think of more elaborate things (like writing a script to compare all gems source against their gem repos prior to release) but it really feels like the biggest problem here is not having an obvious workflow trigger that some work has not been done yet (e.g. syncing gems).

#15 - 02/25/2022 05:40 PM - headius (Charles Nutter)

This happened again in Ruby 3.1. I have not audited the other gems, but at least securerandom was modified in both repositories but those changes were not released as a new gem.

https://github.com/ruby/securerandom/issues/10

#16 - 01/26/2023 07:16 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #19351: Promote bundled gems at Ruby 3.3 added

#17 - 02/08/2023 06:34 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Assigned to Closed

In Ruby 3.2, I completely sync default gems and ruby release code. But Ruby 3.1 and 3.0 are still work in progress. I'll working to fix out-of-sync statuses with @headius (Charles Nutter) on each repository.